CLAIMS

What is claimed is:

1. A method of generating common intermediate language code comprising:

receiving a portion of JAVA^{TM} language source code referencing a first class having a definition that is uniformly applicable to a plurality of classes associated with the first class, the source code identifying one of the plurality of associated classes; and

generating language-neutral intermediate language code representing the portion of source code.

2. A method as recited in claim 1 further comprising parsing the portion of the source code into a parse tree representing the source code.

3. A method as recited in claim 2 further comprising nesting a constructed class of the first class in the parse tree.

4. A method as recited in claim 1 further comprising:

generating a parse tree having a token referencing the first class and a token referencing the specified one of the plurality of associated classes; and

semantically analyzing the parse tree to determine validity of semantics of the first class.

5. A method as recited in claim 4 wherein the semantically analyzing comprises determining whether operations applied to the first class are valid.

6. A method as recited in claim 1 further comprising generating metadata descriptive of the first class.

7. A method as recited in claim 6 further comprising storing the metadata with the language-neutral intermediate language code, whereby the language-neutral intermediate language code may be used by an application program.

8. A method as recited in claim 1 further comprising creating a compiled project including the language-neutral intermediate language code and metadata descriptive of the first class and the specified one of the plurality of associated classes.

9. A method as recited in claim 1 further comprising executing the language-neutral intermediate language code with a runtime engine.

10. A method as recited in claim 1 further comprising developing the portion of source code in a framework that provides the definition of the first class.

11. A method as recited in claim 10 wherein the framework is a .NET™ Framework.

12. A method as recited in claim 11 wherein the developing comprises authoring the portion of source code with a VISUAL J# .NET^{TM} application of the .NET^{TM} Framework.

13. A method of compiling comprising:

receiving a portion of JAVA^{TM} language software having a declaration of an instance of a generic class;

parsing the declaration into a token corresponding to the generic class; and

creating an intermediate language code block corresponding to the parsed declaration, the intermediate language code block executable by a runtime engine.

14. A method as recited in claim 13 further comprising associating the declaration of the instance of the generic class with a defined generic class in a generic class library.

15. A method as recited in claim 14 further comprising tokenizing a parse tree with an identifier corresponding to the defined generic class, the parse tree comprising a hierarchical representation of the declaration.

16. A method as recited in claim 13 further comprising creating metadata describing the portion of the JAVA^{TM} language software.

17. A method as recited in claim 14 further comprising validating an operation on the instance of the generic class based on the defined generic class.

18. A computer-readable medium having stored thereon computer-executable instructions for performing a method of compiling comprising:

receiving a portion of JAVA^{TM} language software including an instruction that references a generic class of a specified type;

creating a parse tree having a generic class identifier associated with the generic class and type identifier associated with the specified type; and

generating one or more intermediate language instructions representing the JAVA^{TM} language instruction based on the parse tree.

19. A computer-readable medium as recited in claim 18, the method further comprising translating the one or more intermediate language instructions into microprocessor-specific binary for execution by a computer.

20. A computer-readable medium as recited in claim 18, the method further comprising validating the parse tree according to a generic class definition associated with the generic class.

21. A computer-readable medium as recited in claim 20, wherein validating the parse tree comprises determining whether an assignment applied to the instance of the generic class assigns an allowable type to the instance.

22. A computer-readable medium as recited in claim 18, the method further comprising generating metadata associated with the generic class.

23. A computer- readable medium as recited in claim 18, wherein the specified type is a second generic class of a second specified type.

24. A computer- readable medium as recited in claim 23, wherein the method further comprises nesting the second generic class and the second specified type at different levels in a hierarchy in the parse tree.

25. A computer-readable medium having stored thereon a data structure for use by a compiler, the data structure comprising:

a generic class identifier field having data identifying a generic class referenced in a portion of source code in a language for which a generic class syntax is not formally specified; and

a constructed class identifier field having data identifying a constructed class of the generic class.

26. A computer-readable medium as recited in claim 25, wherein the data structure further comprises:

at least one nested constructed class that is a generic class.

27. A computer-readable medium as recited in claim 25, wherein the generic class identifier identifies a Queue class.

28. A computer-readable medium as recited in claim 25, wherein the language is a JAVA^{TM} language.

29. A computer-readable medium as recited in claim 25, wherein the data structure further comprises metadata describing the generic class.

30. A computer-readable medium as recited in claim 25, wherein the constructed class comprises one of:

an integer type;

a float type;

a Stack type;

a Queue type; and

a Dictionary type.

31. A method for compiling comprising:

receiving a portion of source code in a language for which .NET^{TM} generic types are not specified in a formal definition of the language;

parsing the portion into a parse tree having an instance of a first type having at least one instance of an associated type; and

generating an intermediate representation of the parse tree.

32. The method as recited in claim 31 further comprising importing metadata describing the first class and the at least one instance of the associated class.

33. The method as recited in claim 31 further comprising tokenizing the parse tree with a token corresponding to the generic type.

34. The method as recited in claim 33 further comprising tokenizing the parse tree with at least one token corresponding to the at least on instance of the associated type.

35. The method as recited in claim 31 wherein the generic type is a .NET^{TM} generic class.

36. A method of generating microprocessor-executable code comprising:

receiving a portion of source code written in a language for which generic classes are unspecified, the portion of source code including a generic class declaration declaring a generic class, the generic class declaration including at least one associated class reference defining a constructed class of the generic class; and

generating a module having microprocessor-executable instructions corresponding to the constructed class, the module further having metadata describing the constructed class.

37. A method as recited in claim 36 wherein the microprocessor-executable instructions comprise intermediate language instructions.

38. A method as recited in claim 36 wherein the microprocessor-executable instructions comprise Microsoft® Intermediate Language instructions.

39. A method as recited in claim 36 wherein the metadata comprises at least one of:

a name of the constructed class;

visibility information indicating the visibility of the constructed class;

inheritance information indicating a class from which the constructed class derives;

interface information indicating one or more interfaces implemented by the constructed class;

method information indicating one or more methods implemented by the constructed class;

properties information indicating identifying at least one property exposed by the constructed class; and

events information indicating at least one event the constructed class provides.

40. A method of compiling comprising:

receiving a portion of source code written in a language for which generic classes are unspecified in a formal language specification, the portion of source code including a first class reference having at least one associated class reference referencing a class associated with the first class; and

generating an intermediate language representation of the portion of source code, the intermediate representation having an instance of the first class and an instance of the at least one associated class.

41. A method as recited in claim 40 wherein the first class is a generic class.

42. A method as recited in claim 40 wherein the language is a JAVA^{TM} language.

43. A method as recited in claim 40 further comprising validating the type based on a definition of the first class.

44. A method as recited in claim 43 further comprising validating an operation on the first class based on a definition of the first class.

45. A method as recited in claim 40 further comprising interpreting the intermediate representation for execution by a microprocessor.

46. A method as recited in claim 40 wherein angular brackets surround the at least one associated class reference.

47. A method as recited in claim 40 wherein the first class is a Queue class.

48. A method as recited in claim 47 wherein the at least one associated class comprises at least one of:

an int type;

a string type; and

a Queue type.

49. A method as recited in claim 48 wherein the Queue type includes at least one nested class reference referencing a second type associated with the Queue type.

50. A method as recited in claim 40 wherein the at least one associated class reference includes one or more nested generic class references.

51. A system for compiling comprising:

a parser receiving JAVA^{TM} language source code having an instruction referencing a generic class and specifying a type of the generic class, the parser further creating a parse tree from the source code, the parse tree including a first node representing the generic class and a second node representing the specified type of the generic class; and

a code generator generating intermediate language code representing the source code.

52. A system as recited in claim 51 further comprising:

a common intermediate language importer providing tokens associated with the generic class and the specified type of the generic class.

53. A system as recited in claim 51 further comprising a runtime engine executing the intermediate language code.

54. A system as recited in claim 51 further comprising a semantic analyzer analyzing the specified type to determine whether the specified type is an allowable type of the generic class.